Research Plan for Matching in Multi Agent Pathfinding using M*

Jonathan Dönszelmann

April 23, 2021

Title:	Matching in Multi Agent Pathfinding using M*
Author:	Jonathan Dönszelmann
Responsible Professor:	Mathijs de Weerdt
Other Supervisor:	Jesse Mulderij
Peer group members:	Robbin Baauw, Thom van der Woude, Ivar de Bruin, Jaap de Jong

Contents

Introduction

This document is a plan for research which I will do during the fourth quarter of study year 2020/2021 about adding matching to multi agent pathfinding. In this plan I will give a definition of a problem, a set of questions I will be answering, and a planning of how these questions will be answered.

1 Background of the research

1.1 Definition of multi agent pathfinding

Stern (2019) [1] defines a multi agent pathfinding (in this plan abbreviated to *MAPF*) as a problem of k so-called agents [1] and the following tuple:

$$\langle G, s, g \rangle$$

[1] Agents can be thought of as robots travelling through a graph

- G is a graph $\langle V, E \rangle$
- V is a set of vertices
- E is a set of edges between vertices
- s is a list of k vertices where every s_i is a starting position for an agent a_i
- g is a list of k vertices where every g_i is a target position for an agent a_i

An algorithm to solve *MAPF* finds solutions such that each agent moves from their starting position to their target position. Each agent a_i has a path π_i. Each path π_i is a list of positions, defining where each agent a_i is at a moment t (referred to as $pi_i[t]$). The cost of a path π_i is its length.
However, while moving, there are a number of conflicts that are to be avoided by the agents.

- Edge conflict: two agents are on the same edge E at a timestep t ($\pi_i[t] = \pi_j[t]$ and $\pi_i[t+1] = \pi_j[t+1]$)
- Vertex conflict: two agents are on the same vertex V at a timestep t ($\pi_i[t] = \pi_j[t]$)
- Swapping: two agents swap vertices at a timestep t ($\pi_i[t] = \pi_j[t+1]$ and $\pi_i[t+1] = \pi_j[t]$)
- Following conflict: an agent travels to a vertex from which another agent leaves in the same timestep ($\pi_i[t+1] = \pi_j[t]$)
- Cycle conflict: a number of agents larger than 2 all swap vertices such that each of the agents lands on a vertex another agent in the set just left.

Summarised, these conflicts define that in *MAPF* it is not allowed for agents to collide in any way. Notable is that some definitions of *MAPF* allow for some of the conflicts defined above to be broken. For example, in previous research that attempted to add waypoints to *MAPF*, the following constraint and cycle constraint were dropped [2]. The reasoning behind this was that when seeing agents as being point-like, and when agents are following each other, they are never actually colliding.

Algorithms solving *MAPF* are minimising either the *makespan*, which is the cost of the longest single path, or the *sum of costs* which is the combined cost of all agents' paths. The cost for a single path is defined to be the number of time steps until the last time an agent ends on its target. This means that waiting also has a cost of 1, unless the agent is waiting on its goal location and won't move again in the simulation.

1.2 Matching

Matching in a bipartite graph is the problem of finding a set of connections between two parts of a graph. No vertex in one part of the graph may be connected to another part of the graph by two edges in the *matching*. Adding *matching* to *MAPF* (in this plan referred to as *MAPFM*) alters the original problem definition in the following way:

$$\langle G, s, g, sc, gc \rangle$$

Two variables are added to the definition of this problem:

- sc is an array of colours sc_i for each starting vertex s_i
- gc is an array of colours gc_i for each target vertex g_i

For a solution to *MAPFM*, it does not matter which agent a_i travels to which target g_i. Instead, an agent a_i can travel to any g_i as long as $sc_i = gc_i$.
However, *MAPFM* is not yet fully defined. For example, is it allowed for there to be more target positions than there are agents? Or conversely, is it allowed for there to be fewer target positions than there are agents? In this research it will be assumed that the number of agents will be equal to the number of goals. However, when time permits, the other two options will also be explored.

In this research, the graph G will be simplified to be a 4-connected grid.

1.3 Algorithms

A number of algorithms exist to solve the *MAPF* problem. These include A* with operator decomposition and independence detection (*A*+OD+ID*) [3], *M** [4], *Conflict based search* [5] and *Branch and cut and price* [6].
However, to solve the *MAPFM*, no algorithm currently exists. It is planned that each of these algorithms will be adapted to be able to solve *MAPFM* through this this research, and through other similar research by peers.

M*

*M**, as previously described, is an algorithm designed to solve *MAPF*. I will briefly outline it's workings as this is the algorithm I will be proposing to adapt to be able to use matching.
*M** solves MAPF by planning paths for each agent individually using *A** pathfinding. However, when it detects that two agents will be in conflict with each other, it will for a short time, plan the paths of these two agents together in such a way that the conflict is resolved. This is very similar to what (*A*+OD+ID*) does. But in (*A*+OD+ID*), when two agents are in conflict their entire path will be planned together, while in *M**, paths can be planned independently again after the conflict.
*M** without extensions, guarantees to generate optimal paths for *MAPF*. However, in prior research extending *MAPF* to allow for waypoints to be added to paths, optimality could no longer be guaranteed (in certain cases). [2]

1.4 Applications and related problems

MAPF is a problem which very naturally applies to railways. Trains moving around shunting yards are like agents. They cannot collide or move past each other as they are bound to the tracks they ride on. But in shunting yards, it is usually less important which exact train moves where, and more important what type of train moves somewhere. For example, if trains are the same length, then it does not matter very much which one goes where in the shunting yard as long as they both end up somewhere with enough space.
When trains are stored and serviced in a shunting yard, they need to be moved around. From their parking space, to servicing stations, and back to service to transport people. The scheduling problem that arises is called the Train unit Shunting problem (here abbreviated as *TUSP*). To find relaxations to the *TUSP* problem, Mulderij used *MAPF* [7]. In his work he describes that extensions to *MAPF* (such as matching) are necessary to make this viable. That is in fact the root of this planned research.

Research Question

During this research, I will be answering the following question:

> Can *M** be adapted to efficiently find optimal solutions to the *MAPFM* problem?

To answer this question, I think I will need to answer the following sub-questions. However, while researching, it is possible that more of these will arise. If so, and if they fall in the scope of the project, they may be added.

- Can *M** be adapted to do matching
- Are *M** solutions still optimal with matching?
- Are there more ways to perform matching, and which way is fastest?
- How does the runtime cost of adding matching to *M** compare to adding matching to other base algorithms?
- Are there heuristics which can improve the runtime of *M** with matching?
- Is it possible to adapt *M** with something similar to operator decomposition.
- What happens when the number of agents is no longer equal to the number of target locations?

The last sub-question I find especially interesting since real life scenarios (with for example shunting yards) will certainly have situations in which the number of agents is not equal to the number of target locations.
The second to last question is an idea I myself had, it feels quite natural to combine the two but have not been able to find any literature on this. I will investigate this.
From now on, I will call the algorithm I will be developing to solve *MAPFM* using *M** as base algorithm *M*MAPFM*

2 Method

In chronological order, this is how I intend to go about researching the previously stated subquestions:

- Implement *M** for plain *MAPF* problems to get a starting point which can be extended to do matching.
- Extend this *MAPF* solver to also solve *MAPFM*.
- Create a number of benchmarks to quantitatively evaluate the performance of this algorithm.
- Compare *M** to the algorithms developed by peers by running them on these same benchmarks.
- Research ways to improve the runtime of *M*MAPFM*, for example by using better heuristics.
- Research possibilities to allow for a different number of agents than there are

To fairly compare benchmarks, I will be working on an online system to compare *MAPFM* algorithms, adapted from a similar website used in 2020 by a group researching waypoints in multi agent pathfinding. This will not be a part of the research, but I believe that it will be very valuable when comparing our algorithms.

3 Planning of the research project

Planning

week 1 (ma 19 apr)	• Kick off • Lecture Research methods • Create planning for week 1 (Monday) • Assignment Information literacy (Tuesday) • Distribute base algorithms • Create research plan • Benchmarking website • Meeting with supervisor (Thursday)
week 2 (ma 26 apr)	• Prepare presentation of research plan. • Presentation of research plan (Thursday) • Start implementing base algorithm • Lecture responsible research • Meeting with supervisor (Thursday)
week 3 (ma 3 may)	• Lecture Academic communication skills (Friday) • Implement base algorithm • Research extending to *MAPFM* • Meeting with supervisor (Thursday)
week 4 (ma 10 may)	• Session responsible research (Monday) • Meeting with supervisor (Thursday) • Extend the algorithm to be able to solve *MAPFM*.
week 5 (ma 17 may)	• Session Academic communication skills (Monday) • Prepare midterm presentation (before Wednesday) • Midterm presentation (Wednesday) • Meeting with supervisor (Thursday) • Create benchmarks and start comparing to other algorithms (made by peers).

week 6 (ma 24 may)	• Meeting with supervisor (Thursday) • Session Academic communication skills (Friday) • Find ways to improve runtime performance of *M*MAPF* • Start working on the basics of the final paper.
week 7 (ma 31 may)	• Quantitative evaluation and comparison with other algorithms. • Possibly extend to solve *MAPFM* with a different number of agents than targets. • Work on paper for draft deadline. • Meeting with supervisor (Thursday)
week 8 (ma 7 jun)	• Paper draft v1 (Monday) • Peer review paper draft v1 (Thursday) • Meeting with supervisor (Thursday) • Improve paper for draft 2. • Find more ways to improve *M*MAPFM*
week 9 (ma 14 jun)	• Paper draft v2 (Wednesday) • Final work on paper.
week 10 (ma 21 jun)	• Submit final paper version (day tbd?) • Create poster and work on poster presentation. • Meeting with supervisor (Thursday)
week 11 (ma 28 jun)	• Session Academic communication skills (Monday) • submission poster (Tuesday) • poster presentation (Thursday or Friday) • Meeting with examiner

References

[1] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *arXiv preprint arXiv:1906.08291*, 2019.

[2] J. van Dijk, "Solving the multi-agent path finding with waypoints problem using subdimensional expansion," 2020.

[3] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010.

[4] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2011, pp. 3260–3267.

[5] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[6] E. Lam, P. Le Bodic, D. D. Harabor, and P. J. Stuckey, "Branch-and-cut-and-price for multi-agent pathfinding.," in *IJCAI*, 2019, pp. 1289–1296.

[7] J. Mulderij, B. Huisman, D. Tönissen, K. van der Linden, and M. de Weerdt, "Train unit shunting and servicing: A real-life application of multi-agent path finding," *arXiv preprint arXiv:2006.10422*, 2020.